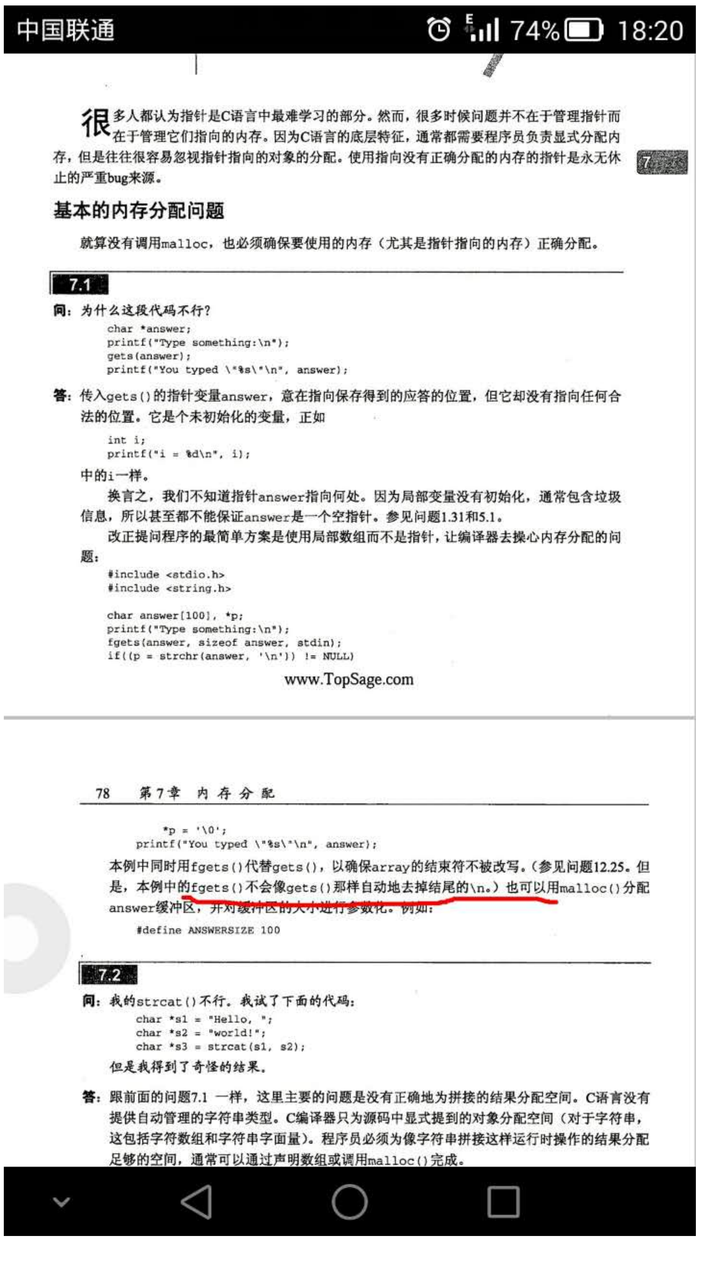
####置顶

####笔记
嵌入式Linux C语言开发 笔记
你必须知道的495个c语言问题 笔记
程序员面试宝典~~笔记
c和指针 笔记

==========updating==========

####几个常见算法
1. 分治算法
分治法的设计思想：将一个难以直接解决的大问题，分割成一些规模较小的相同问题，以便各个击破，分而治之。
分治法的策略：对于一个规模为n的问题，若该问题可以容易地解决（比如说规模n较小）则直接解决，否则将其分解为k个规模较小的子问题，这些子问题互相独立且与原问题形式相同，递归地解这些子问题，然后将各子问题的解合并得到原问题的解。
分治法所能解决的问题一般具有以下几个特征：
1) 该问题的规模缩小到一定的程度就可以容易地解决
2) 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质。
3) 利用该问题分解出的子问题的解可以合并为该问题的解；
4) 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子子问题。
第一条特征是绝大多数问题都可以满足的，因为问题的计算复杂性一般是随着问题规模的增加而增加；
第二条特征是应用分治法的前提它也是大多数问题可以满足的，此特征反映了递归思想的应用；、
第三条特征是关键，能否利用分治法完全取决于问题是否具有第三条特征，如果具备了第一条和第二条特征，而不具备第三条特征，则可以考虑用贪心法或动态规划法。
第四条特征涉及到分治法的效率，如果各子问题是不独立的则分治法要做许多不必要的工作，重复地解公共的子问题，此时虽然可用分治法，但一般用动态规划法较好。
可使用分治法求解的一些经典问题：
合并排序、快速排序、二分搜索、大整数乘法、Strassen矩阵乘法、棋盘覆盖、线性时间选择、最接近点对问题、循环赛日程表、汉诺塔
2. 动态规划
基本思想与分治法类似，也是将待求解的问题分解为若干个子问题（阶段），按顺序求解子阶段，前一子问题的解，为后一子问题的求解提供了有用的信息。在求解任一子问题时，列出各种可能的局部解，通过决策保留那些有可能达到最优的局部解，丢弃其他局部解。依次解决各子问题，最后一个子问题就是初始问题的解。
与分治法最大的差别是：适合于用动态规划法求解的问题，经分解后得到的子问题往往不是互相独立的（即下一个子阶段的求解是建立在上一个子阶段的解的基础上，进行进一步的求解）。
适用的情况
能采用动态规划求解的问题的一般要具有3个性质：
1) 最优化原理：如果问题的最优解所包含的子问题的解也是最优的，就称该问题具有最优子结构，即满足最优化原理。
2) 无后效性：即某阶段状态一旦确定，就不受这个状态以后决策的影响。也就是说，某状态以后的过程不会影响以前的状态，只与当前状态有关。
3) 有重叠子问题：即子问题之间是不独立的，一个子问题在下一阶段决策中可能被多次使用到。（该性质并不是动态规划适用的必要条件，但是如果没有这条性质，动态规划算法同其他算法相比就不具备优势）
3. 贪心算法
贪心算法总是作出在当前看来最好的选择。也就是说贪心算法并不从整体最优考虑，它所作出的选择只是在某种意义上的局部最优选择。当然，希望贪心算法得到的最终结果也是整体最优的。虽然贪心算法不能对所有问题都得到整体最优解，但对许多问题它能产生整体最优解。如单源最短路经问题，最小生成树问题等。在一些情况下，即使贪心算法不能得到整体最优解，其最终结果却是最优解的很好近似。
从许多可以用贪心算法求解的问题中看到这类问题一般具有2个重要的性质：贪心选择性质和最优子结构性质。
贪心选择性质是指所求问题的整体最优解可以通过一系列局部最优的选择，即贪心选择来达到。这是贪心算法可行的第一个基本要素，也是贪心算法与动态规划算法的主要区别。
当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。贪心算法和动态规划算法都要求问题具有最优子结构性质，这是两个算法的一个共同点。
例如：活动安排问题、背包问题
4. 回溯法
回溯法的基本做法是搜索，或是一种组织得井井有条的，能避免不必要搜索的穷举式搜索法。这种方法适用于解一些组合数相当大的问题。
回溯法在问题的解空间树中，按深度优先策略，从根结点出发搜索解空间树。算法搜索至解空间树的任意一点时，先判断该结点是否包含问题的解。如果肯定不包含（剪枝过程），则跳过对该结点为根的子树的搜索，逐层向其祖先结点回溯；否则，进入该子树，继续按深度优先策略搜索。
5. 分支限界法
类似于回溯法，也是一种在问题的解空间树T上搜索问题解的算法。但在一般情况下，分支限界法与回溯法的求解目标不同。回溯法的求解目标是找出T中满足约束条件的所有解，而分支限界法的求解目标则是找出满足约束条件的一个解，或是在满足约束条件的解中找出使某一目标函数值达到极大或极小的解，即在某种意义下的最优解。
所谓“分支”就是采用广度优先的策略，依次搜索E-结点的所有分支，也就是所有相邻结点，抛弃不满足约束条件的结点，其余结点加入活结点表。然后从表中选择一个结点作为下一个E-结点，继续搜索。
由于求解目标不同，导致分支限界法与回溯法在解空间树T上的搜索方式也不相同。回溯法以深度优先的方式搜索解空间树T，而分支限界法则以广度优先或以最小耗费优先的方式搜索解空间树T。
分支限界法的搜索策略是：在扩展结点处，先生成其所有的儿子结点（分支），然后再从当前的活结点表中选择下一个扩展对点。为了有效地选择下一扩展结点，以加速搜索的进程，在每一活结点处，计算一个函数值（限界），并根据这些已计算出的函数值，从当前活结点表中选择一个最有利的结点作为扩展结点，使搜索朝着解空间树上有最优解的分支推进，以便尽快地找出一个最优解。
分支限界法常以广度优先或以最小耗费（最大效益）优先的方式搜索问题的解空间树。问题的解空间树是表示问题解空间的一棵有序树，常见的有子集树和排列树。


中国联通 74% 18:20

很多人都认为指针是C语言中最难学习的部分。然而，很多时候问题并不在于管理指针而在于管理它们指向的内存。因为C语言的底层特征，通常都需要程序员负责显式分配内存，但是往往很容易忽视指针指向的对象的分配。使用指向没有正确分配的内存的指针是永无休止的严重bug来源。

基本的内存分配问题

就算没有调用malloc，也必须确保要使用的内存（尤其是指针指向的内存）正确分配。

7.1

问： 为什么这段代码不行？

```
char *answer;
printf("Type something:\n");
gets(answer);
printf("You typed \"%s\"\n", answer);
```

答： 传入gets()的指针变量answer，意在指向保存得到的应答的位置，但它却没有指向任何合法的位置。它是个未初始化的变量，正如

```
int i;
printf("i = %d\n", i);
```

中的i一样。

换言之，我们不知道指针answer指向何处。因为局部变量没有初始化，通常包含垃圾信息，所以甚至都不能保证answer是一个空指针。参见问题1.31和5.1。

改正提问程序的最简单方案是使用局部数组而不是指针，让编译器去操心内存分配的问题：

```
#include <stdio.h>
#include <string.h>

char answer[100], *p;
printf("Type something:\n");
fgets(answer, sizeof answer, stdin);
if((p = strchr(answer, '\n')) != NULL)
```

www.TopSage.com


```
    *p = '\0';
printf("You typed \"%s\"\n", answer);
```

本例中同时用fgets()代替gets()，以确保array的结束符不被改写。（参见问题12.25。但是，本例中的fgets()不会像gets()那样自动地去掉结尾的\n。）也可以用malloc()分配answer缓冲区，并对缓冲区的大小进行参数化。例如：

```
#define ANSWERSIZE 100
```

7.2

问： 我的strcat()不行。我试了下面的代码：

```
char *s1 = "Hello, ";
char *s2 = "world!";
char *s3 = strcat(s1, s2);
```

但是我得到了奇怪的结果。

答： 跟前面的问题7.1 一样，这里主要的问题是没有正确地为拼接的结果分配空间。C语言没有提供自动管理的字符串类型。C编译器只为源码中显式提到的对象分配空间（对于字符串，这包括字符数组和字符串字面量）。程序员必须为像字符串拼接这样运行时操作的结果分配足够的空间，通常可以通过声明数组或调用malloc()完成。